k-Shape: Efficient and Accurate Clustering of Time Series

John Paparrizos
Columbia University
jopa@cs.columbia.edu

Luis Gravano
Columbia University
gravano@cs.columbia.edu

ABSTRACT

The proliferation and ubiquity of temporal data across many disciplines has generated substantial interest in the analysis and mining of time series. Clustering is one of the most popular data mining methods, not only due to its exploratory power, but also as a preprocessing step or subroutine for other techniques. In this paper, we present k-Shape, a novel algorithm for time-series clustering. k-Shape relies on a scalable iterative refinement procedure, which creates homogeneous and well-separated clusters. As its distance measure, k-Shape uses a normalized version of the cross-correlation measure in order to consider the shapes of time series while comparing them. Based on the properties of that distance measure, we develop a method to compute cluster centroids, which are used in every iteration to update the assignment of time series to clusters. To demonstrate the robustness of k-Shape, we perform an extensive experimental evaluation of our approach against partitional, hierarchical, and spectral clustering methods, with combinations of the most competitive distance measures. k-Shape outperforms all scalable approaches in terms of accuracy. Furthermore, k-Shape also outperforms all non-scalable (and hence impractical) combinations, with one exception that achieves similar accuracy results. However, unlike k-Shape, this combination requires tuning of its distance measure and is two orders of magnitude slower than k-Shape. Overall, k-Shape emerges as a domain-independent, highly accurate, and highly efficient clustering approach for time series with broad applications.

1. INTRODUCTION

Temporal, or sequential, data mining deals with problems where data are naturally organized in sequences [34]. We refer to such data sequences as time-series sequences if they contain explicit information about timing (e.g., stock, audio, speech, and video) or if an ordering on values can be inferred (e.g., streams and handwriting). Large volumes of time-series sequences appear in almost every discipline, including astronomy, biology, meteorology, medicine, finance, robotics, engineering, and others [1, 6, 25, 27, 36, 52, 70, 76]. The ubiquity of time series has generated a substantial interest in querying [2, 19, 45, 46, 48, 62, 74, 79], indexing [9, 13, 41, 42, 44, 77], classification [37, 56, 68, 88], clustering [43, 54, 64, 87, 89], and modeling [4, 38, 86] of such data.

SIGMOD '15, May 31–June 4, 2015, Melbourne, Victoria, Australia.

http://dx.doi.org/10.1145/2723372.2737793.

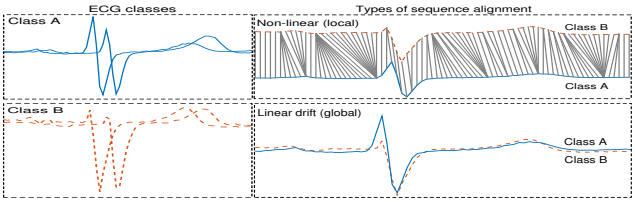


Figure 1: ECG sequence examples and types of alignments for the two classes of the ECGFiveDays dataset [1].

Among all techniques applied to time-series data, clustering is the most widely used as it does not rely on costly human supervision or time-consuming annotation of data. With clustering, we can identify and summarize interesting patterns and correlations in the underlying data [33]. In the last few decades, clustering of time-series sequences has received significant attention [5, 16, 25, 47, 60, 64, 66, 87, 89], not only as a powerful stand-alone exploratory method, but also as a preprocessing step or subroutine for other tasks.

Most time-series analysis techniques, including clustering, critically depend on the choice of distance measure. A key issue when comparing two time-series sequences is how to handle the variety of distortions, as we will discuss, that are characteristic of the sequences. To illustrate this point, consider the well-known ECGFiveDays dataset [1], with ECG sequences recorded for the same patient on two different days. While the sequences seem similar overall, they exhibit patterns that belong in one of the two distinct classes (see Figure 1): Class A is characterized by a sharp rise, a drop, and another gradual increase while Class B is characterized by a gradual increase, a drop, and another gradual increase. Ideally, a *shape-based* clustering method should generate a partition similar to the classes shown in Figure 1, where sequences exhibiting similar patterns are placed into the same cluster based on their *shape* similarity, regardless of differences in amplitude and phase. As the notion of shape cannot be precisely defined, dozens of distance measures have been proposed [11, 12, 14, 19, 20, 22, 55, 75, 78, 81] to offer invariances to multiple inherent distortions in the data. However, it has been shown that distance measures offering invariances to amplitude and phase perform exceptionally well [19, 81] and, hence, such distance measures are used for shape-based clustering [53, 59, 64, 87].

Due to these difficulties and the different needs for invariances from one domain to another, more attention has been given to the creation of new distance measures rather than

to the creation of new clustering algorithms. It is generally believed that the choice of distance measure is more important than the clustering algorithm itself [7]. As a consequence, time-series clustering relies mostly on classic clustering methods, either by replacing the default distance measure with one that is more appropriate for time series, or by transforming time series into "flat" data so that existing clustering algorithms can be directly used [83]. However, the choice of clustering method can affect: (i) accuracy, as every method expresses homogeneity and separation of clusters differently; and (ii) efficiency, as the computational cost differs from one method to another. For example, spectral clustering [21] or certain variants of hierarchical clustering [40] are more appropriate to identify density-based clusters (i.e., areas of higher density than the remainder of the data) than partitional methods such as k-means [50] or k-medoids [40]. On the other hand, k-means is more efficient than hierarchical, spectral, or k-medoids methods.

Unfortunately, state-of-the-art approaches for shape-based clustering, which use partitional methods with distance measures that are scale- and shift-invariant, suffer from two main drawbacks: (i) these approaches cannot scale to large-volumes of data as they depend on computationally expensive methods or distance measures [53, 59, 64, 87]; (ii) these approaches have been developed for particular domains [87] or their effectiveness has only been shown for a limited number of datasets [53, 59]. Moreover, the most successful shape-based clustering methods handle phase invariance through a local, non-linear alignment of the sequence coordinates, even though a global alignment is often adequate. For example, for the ECG dataset in Figure 1, an efficient linear drift can reveal the underlying differences in patterns of sequences of two classes, whereas an expensive non-linear alignment might match every corresponding increase or drop of each sequence, making it difficult to distinguish the two classes (see Figure 1). Importantly, to the best of our knowledge, these approaches have never been extensively evaluated against each other, against other partitional methods, or against different approaches such as hierarchical or spectral methods. We present such an experimental evaluation in this paper, as discussed below.

In this paper, we propose k-Shape, a novel algorithm for shape-based time-series clustering that is efficient and domain independent. k-Shape is based on a scalable iterative refinement procedure similar to the one used by the k-means algorithm, but with significant differences. Specifically, k-Shape uses both a different distance measure and a different method for centroid computation from those of k-means. As argued above, k-Shape attempts to preserve the shapes of time-series sequences while comparing them. To do so, k-Shape requires a distance measure that is invariant to scaling and shifting. Unlike other clustering approaches [53, 64, 87], for k-Shape we adapt the cross-correlation statistical measure and we show: (i) how we can derive in a principled manner a time-series distance measure that is scale- and shift-invariant; and (ii) how this distance measure can be computed efficiently. Based on the properties of the normalized version of cross-correlation, we develop a novel method to compute cluster centroids, which are used in every iteration to update the assignment of time series to clusters.

To demonstrate the effectiveness of the distance measure and k-Shape, we have conducted an extensive experimental evaluation on 48 datasets and compared the state-of-the-art distance measures and clustering approaches for time series using rigorous statistical analysis. We took steps to ensure the reproducibility of our results, including making available our source code as well as using public datasets. Our results show that our distance measure is competitive, outperforming Euclidean distance (ED) [20], and achieving similar results as constrained Dynamic Time Warping (cDTW) [72], the best performing distance measure [19, 81], without requiring any tuning and performing one order of magnitude faster. For example, for the ECG dataset in Figure 1, our distance measure achieves a one-nearest-neighbor classification accuracy of 98.9%, significantly higher than cDTW's accuracy for the task, namely, 79.7%.

For clustering, we show that the k-means algorithm with ED, in contrast to what has been reported in the literature, is a robust approach and that inadequate modifications of the distance measure and the centroid computation can reduce its performance. Moreover, simple partitional methods outperform hierarchical and spectral methods with the most competitive distance measures, indicating that the choice of algorithm, which is sometimes believed to be less important than that of distance measure, is as critical as the choice of distance measure. Similarly, we show that k-Shape outperforms all scalable and non-scalable partitional, hierarchical, and spectral methods in terms of accuracy, with the only exception of one existing approach that achieves similar results, namely, k-medoids with cDTW. However, there are problems with this approach that can be avoided with k-Shape: (i) the requirement of k-medoids to compute the dissimilarity matrix makes it unable to scale and particularly slow, two orders of magnitude slower than k-Shape; (ii) its distance measure requires tuning, either through automated methods that rely on labeling of instances or through the help of a domain expert; this requirement is problematic for clustering, which is an unsupervised task. In contrast, k-Shape uses a simple, parameter-free distance measure. Overall, k-Shape is a highly accurate and scalable choice for time-series clustering that performs exceptionally well across different domains. k-Shape is particularly effective for applications involving similar but out-of-phase sequences, such as those of the ECG dataset in Figure 1, for which k-Shape reaches an 84% clustering accuracy, which is significantly higher than the 53% accuracy for k-medoids with cDTW.

In this paper, we start with an in-depth review of the state of the art for clustering time series, as well as with a precise definition of our problem of focus (Section 2). We then present our novel approach as follows:

- We show how a scale-, translate-, and shift-invariant distance measure can be derived in a principled manner from the cross-correlation measure and how this measure can be efficiently computed (Section 3.1).
- We present a novel method to compute a cluster centroid when that distance measure is used (Section 3.2).
- We develop k-Shape, a centroid-based algorithm for time-series clustering (Section 3.3).
- We evaluate our ideas by conducting an extensive experimental evaluation (Sections 4 and 5).

We conclude with a discussion of related work (Section 6) and the implications of our work (Section 7).

2. PRELIMINARIES

In this section, we review the relevant theoretical background (Section 2.1). We discuss distortions that are common in time series (Section 2.2) and the most popular distance measures for such data (Section 2.3). Then, we summarize existing approaches for clustering time-series data (Section 2.4) and for centroid computation (Section 2.5). Finally, we formally present our problem of focus (Section 2.6).

2.1 Theoretical Background

Hardness of clustering: Clustering is the general problem of partitioning n observations into k clusters, where a *cluster* is characterized with the notions of homogeneity — the similarity of observations within a cluster — and separation — the dissimilarity of observations from different clusters. Even though many clustering criteria to capture homogeneity and separation have been proposed [35], the minimum within-cluster sum of squared distances is most commonly used as it expresses both of them. Given a set of n observations $X = \{\vec{x}_1, \ldots, \vec{x}_n\}$, where $\vec{x}_i \in \mathbb{R}^m$, and the number of clusters $k < n$, the objective is to partition X into k pairwise-disjoint clusters $P = \{p_1, \ldots, p_k\}$, such that the within-cluster sum of squared distances is minimized:

$$P^* = \arg\min_{P} \sum_{j=1}^{k} \sum_{\vec{x}_i \in p_j} dist(\vec{x}_i, \vec{c}_j)^2 \qquad (1)$$

where $\vec{c}_j$ is the centroid of partition $p_j \in P$. In Euclidean space this is an NP-hard optimization problem for $k \geq 2$ [3], even for number of dimensions $m = 2$ [51]. Because finding a global optimum is difficult, heuristics such as the k-means method [50] are often used to find a local optimum. Specifically, k-means randomly assigns the data points into k clusters and then uses an iterative procedure that performs two steps in every iteration: (i) in the assignment step, every data point is assigned to the cluster of its nearest centroid, which is determined with the use of a distance function; (ii) in the refinement step, the centroids of the clusters are updated to reflect the changes in cluster memberships. The algorithm converges either when there is no change in cluster memberships or when the maximum number of iterations is reached.

Steiner's sequence: In the refinement step, k-means computes new centroids to serve as representatives of the clusters. The centroid is defined as the data point that minimizes the sum of squared distances to all other data points and, hence, it depends on the distance measure used. Finding such a centroid is known as the Steiner's sequence problem [63]: given a partition $p_j \in P$, the corresponding centroid $\vec{c}_j$ needs to fulfill:

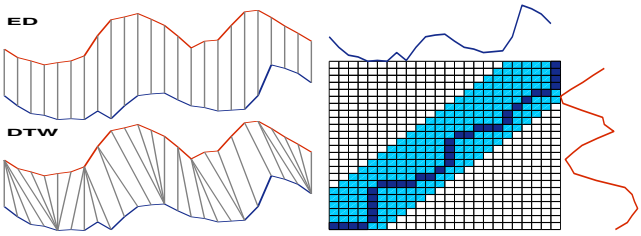
$$\vec{c}_j = \arg\min_{\vec{w}} \sum_{\vec{x}_i \in p_j} dist(\vec{w}, \vec{x}_i)^2, \quad \vec{w} \in \mathbb{R}^m \qquad (2)$$

When ED is used, the centroid can be computed with the arithmetic mean property [18]. In many cases where alignment of observations is required, this problem is referred to as the multiple sequence alignment problem, which is known to be NP-complete [80]. In the context of time series, Dynamic Time Warping (DTW) (see Section 2.3) is the most widely used measure to compare time-series sequences with alignment, and many heuristics have been proposed to find the average sequence under DTW (see Section 2.5).

2.2 Time-Series Invariances

Based on the domain, sequences are often distorted in some way, and distance measures need to satisfy a number of invariances in order to compare sequences meaningfully. In this section, we review common time-series distortions and their invariances. For a more detailed review, see [7].

Scaling and translation invariances: In many cases, it is useful to recognize the similarity of sequences despite differences in amplitude (scaling) and offset (translation). In other words, transforming a sequence $\vec{x}$ as $\vec{x}' = a\vec{x} + b$, where a and b are constants, should not change $\vec{x}$'s similarity to other sequences. For example, these invariances might be useful to analyze seasonal variations in currency values on foreign exchange markets without being biased by inflation.



(a) (b)

Figure 2: Similarity computation: (a) alignment under ED (top) and DTW (bottom), (b) Sakoe-Chiba band with a warping window of 5 cells (light cells in band) and the warping path computed under cDTW (dark cells in band).

Shift invariance: When two sequences are similar but differ in phase (global alignment) or when there are regions of the sequences that are aligned and others are not (local alignment), we might still need to consider them similar. For example, heartbeats can be out of phase depending on when we start taking the measurements (global alignment) and handwritings of a phrase from different people will need alignment depending on the size of the letters and on the spaces between words (local alignment).

Uniform scaling invariance: Sequences that differ in length require either stretching of the shorter sequence or shrinking of the longer sequence so that we can compare them effectively. For example, this invariance is required for heartbeats with measurement periods of different duration.

Occlusion invariance: When subsequences are missing, we can still compare the sequences by ignoring the subsequences that do not match well. This invariance is useful in handwritings if there is a typo or a letter is missing.

Complexity invariance: When sequences have similar shape but different complexities, we might want to make them have low or high similarity based on the application. For example, audio signals that were recorded indoors and outdoors might be considered similar, despite the fact that outdoor signals will be more noisy than indoor signals.

For many tasks, some or all of the above invariances are required when we compare time-series sequences. To satisfy the appropriate invariances, we could preprocess the data to eliminate the corresponding distortions before clustering. For example, by z-normalizing [29] the data we can achieve the scaling and translation invariances. However, for invariances that cannot be trivially achieved with a preprocessing step, we can define sophisticated distance measures that offer distortion invariances. In the next section, we review the most common such distance measures.

2.3 Time-Series Distance Measures

The two state-of-the-art approaches for time-series comparison first z-normalize the sequences and then use a distance measure to determine their similarity, and possibly capture more invariances. The most widely used distance metric is the simple ED [20]. ED compares two time series $\vec{x} = (x_1, \ldots, x_m)$ and $\vec{y} = (y_1, \ldots, y_m)$ of length m as follows:

$$ED(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \qquad (3)$$

Another popular distance measure is DTW [72]. DTW can be seen as an extension of ED that offers a local (non-linear) alignment. To achieve that, an m-by-m matrix M is constructed, with the ED between any two points of $\vec{x}$ and $\vec{y}$. A

warping path $W = \{w_1, w_2, \ldots, w_k\}$, with $k \geq m$, is a contiguous set of matrix elements that defines a mapping between $\vec{x}$ and $\vec{y}$ under several constraints [44]:

$$DTW(\vec{x}, \vec{y}) = \min \sqrt{\sum_{i=1}^{k} w_i} \quad (4)$$

This path can be computed on matrix M with dynamic programming for the evaluation of the following recurrence:

$$\gamma(i,j) = ED(i,j) + \min\{\gamma(i-1,j-1), \gamma(i-1,j), \gamma(i,j-1)\}.$$

It is common practice to constrain the warping path to visit only a subset of cells on matrix M. The shape of the subset matrix is called *band* and the width of the band is called *warping window*. The most frequently used band for constrained Dynamic Time Warping (cDTW) is the Sakoe-Chiba band [72]. Figure 2a shows the difference in alignments of two sequences offered by ED and DTW distance measures, whereas Figure 2b presents the computation of the warping path (dark cells) for cDTW constrained by the Sakoe-Chiba band with width 5 cells (light cells).

Recently, Wang et al. [81] extensively evaluated 9 distance measures and several variants thereof. They found that ED is the most efficient measure with a reasonably high accuracy, and that DTW and cDTW perform exceptionally well in comparison to other measures. cDTW is slightly better than DTW and significantly reduces the computation time. Several optimizations have been proposed to further speed up cDTW [65]. In the next section, we review clustering algorithms that can utilize these distance measures.

2.4 Time-Series Clustering Algorithms

Several methods have been proposed to cluster time series. All approaches generally modify existing algorithms, either by replacing the default distance measures with a version that is more suitable for comparing time series (raw-based methods), or by transforming the sequences into "flat" data so that they can be directly used in classic algorithms (feature- and model-based methods) [83]. Raw-based approaches can easily leverage the vast literature on distance measures (see Section 2.3), which has shown that invariances offered by certain measures, such as DTW, are general and, hence, suitable for almost every domain [19]. In contrast, feature- and model-based approaches are usually domain-dependent and applications on different domains require that we modify the features or models. Because of these drawbacks of feature- and model-based methods, in this paper we follow a raw-based approach.

The three most popular raw-based methods are agglomerative hierarchical, spectral, and partitional clustering [7]. For hierarchical clustering, the most widely used "linkage" criteria are the single, average, and complete linkage variants [40]. Spectral clustering [58] has recently started receiving attention [7] due to its success over other types of data [21]. Among partitional methods, k-means [50] and k-medoids [40] are the most representative examples. When partitional methods use distance measures that offer invariances to scaling, translation, and shifting, we consider them as shape-based approaches. From these methods, k-medoids is usually preferred [83]: unlike k-means, k-medoids computes the dissimilarity matrix of all data sequences and uses actual sequences as cluster centroids; in contrast, k-means requires the computation of artificial sequences as centroids, which hinders the easy adaptation of distance measures other than ED (see Section 2.1). However, from all these methods, only the k-means class of algorithms can scale linearly with the size of the datasets. Recently, k-means was modified to work with: (i) the DTW distance measure [64] and (ii) a distance measure that offers pairwise scaling and shifting of time-series sequences [87]. Both of these modifications rely on new approaches to compute cluster centroids that we will review next.

2.5 Time-Series Averaging Techniques

The computation of an average sequence or, in the context of clustering, a centroid, is a difficult task and it critically depends on the distance measure used to compare time series (see Section 2.1). We now review the state-of-the-art methods for the computation of an average sequence.

With Euclidean distance, the property of arithmetic mean is used to compute an average sequence (e.g., as is the case in the centroid computation of the k-means algorithm). However, as DTW is more appropriate for many time-series tasks [44, 65], several methods have been proposed to average time-series sequences under DTW. Nonlinear alignment and averaging filters (NLAAF) [32] uses a simple pairwise method where each coordinate of the average sequence is calculated as the center of the mapping produced by DTW. This method is applied sequentially to pairs of sequences until only one pair is left. Prioritized shape averaging (PSA) [59] uses a hierarchical method to average sequences. The coordinates of an average sequence are computed as the weighted center of the coordinates of two time-series sequences that were coupled by DTW. Initially, all sequences have weight one, and each average sequence produced in the nodes of the tree has a weight that corresponds to the number of sequences it averages. To avoid the high computation cost of previous approaches, Ranking Shape-based Template Matching Framework (RSTMF) [53] approximates an ordering of the time-series sequences by looking at the distances of sequences to all other cluster centroids, instead of computing the distances of all pairs of sequences.

Several drawbacks of these methods have led to the creation of a more robust technique called Dynamic Time Warping Barycenter Averaging (DBA) [64], which iteratively refines the coordinates of a sequence initially picked from the data. Each coordinate of the average sequence is updated with the use of barycenter[1] of one or more coordinates of the other sequences that were associated with the use of DTW. Among all these methods, DBA seems to be the most efficient and accurate averaging approach when DTW is used [64]. Another averaging technique that is based on matrix decomposition was proposed as part of K-Spectral Centroid Clustering (KSC) [87], to compute the centroid of a cluster when a distance measure for pairwise scaling and shifting is used. In our approach, which we will present in Section 3, we also rely on matrix decomposition to compute centroids.

2.6 Problem Definition

We address the problem of domain-independent, accurate, and scalable clustering of time series into k clusters, for a given value of the target number of clusters k.[2] Even though different domains might require different invariances to data distortions (see Section 2.2), we focus on distance measures that offer invariances to scaling and shifting, which are generally sufficient (see Section 2.3) [19]. Furthermore, to easily adopt such distance measures, we focus our analysis on raw-based clustering approaches, as we argued in Section 2.4. Next, we introduce k-Shape, our novel clustering algorithm.

[1] Barycenter is defined as $b(X_1, \ldots, X_a) = \frac{X_1 + \ldots + X_a}{a}$, where the sums in the numerator are vector additions.

[2] Although the exact estimation of k is difficult without a gold standard, we can do so by varying k and evaluating clustering quality with criteria that capture information intrinsic to the data alone [40].

3. k-SHAPE CLUSTERING ALGORITHM

Our objective is to develop a domain-independent, accurate, and scalable algorithm for time-series clustering, with a distance measure that is invariant to scaling and shifting. We propose k-Shape, a novel centroid-based clustering algorithm that can preserve the shapes of time-series sequences. Specifically, we first discuss our distance measure, which is based on the cross-correlation measure (Section 3.1). Based on this distance measure, we propose a method to compute centroids of time-series clusters (Section 3.2). Finally, we describe our k-Shape clustering algorithm, which relies on an iterative refinement procedure that scales linearly in the number of sequences and generates homogeneous and well-separated clusters (Section 3.3).

3.1 Time-Series Shape Similarity

As discussed earlier, capturing shape-based similarity requires distance measures that can handle distortions in amplitude and phase. Unfortunately, the best performing distance measures offering invariances to these distortions, such as DTW, are computationally expensive (see Section 2.3). To circumvent this efficiency limitation, we adopt a normalized version of the cross-correlation measure.

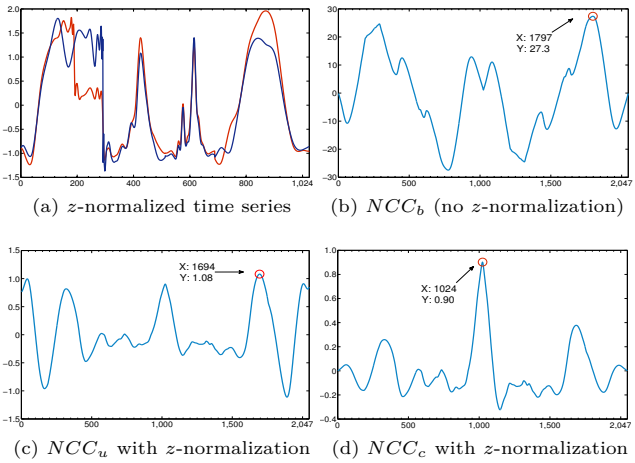
Cross-correlation is a measure of similarity for time-lagged signals that is widely used for signal and image processing. However, cross-correlation, a measure that compares one-to-one points between signals, has largely been ignored in experimental evaluations for the problem of time-series comparison. Instead, starting with the application of DTW decades ago [8], research on that problem has focused on elastic distance measures that compare one-to-many or one-to-none points [11, 12, 44, 55, 78]. In particular, recent comprehensive and independent experimental evaluations of state-of-the-art distance measures for time-series comparison — 9 measures and their variants in [19, 81] and 48 measures in [26] — did not consider cross-correlation. Different needs from one domain or application to another hinder the process of finding appropriate normalizations for the data and the cross-correlation measure. Moreover, inefficient implementations of cross-correlation can make it appear as slow as DTW. As a consequence of these drawbacks, cross-correlation has not been widely adopted as a time-series distance measure. In the rest of this section, we show how to address these drawbacks. Specifically, we will show how to choose normalizations that are domain-independent and efficient, and lead to a shape-based distance measure for comparing time series efficiently and effectively.

Cross-correlation measure: Cross-correlation is a statistical measure with which we can determine the similarity of two sequences $\vec{x} = (x_1, \ldots, x_m)$ and $\vec{y} = (y_1, \ldots, y_m)$, even if they are not properly aligned.[3] To achieve shift-invariance, cross-correlation keeps $\vec{y}$ static and slides $\vec{x}$ over $\vec{y}$ to compute their inner product for each *shift* s of $\vec{x}$. We denote a shift of a sequence as follows:

$$\vec{x}_{(s)} = \begin{cases} (\overbrace{0, \ldots, 0}^{|s|}, x_1, x_2, \ldots, x_{m-s}), & s \geq 0 \\ (x_{1-s}, \ldots, x_{m-1}, x_m, \underbrace{0, \ldots, 0}_{|s|}), & s < 0 \end{cases} \quad (5)$$

When all possible shifts $\vec{x}_{(s)}$ are considered, with $s \in [-m, m]$, we produce $CC_w(\vec{x}, \vec{y}) = (c_1, \ldots, c_w)$, the cross-correlation sequence with length $2m-1$, defined as follows:

[3]For simplicity, we consider sequences of equal length even though cross-correlation can be computed on sequences of different length.



(a) z-normalized time series (b) NCC_b (no z-normalization)

(c) NCC_u with z-normalization (d) NCC_c with z-normalization

Figure 3: Time-series and cross-correlation normalizations.

$$CC_w(\vec{x}, \vec{y}) = R_{w-m}(\vec{x}, \vec{y}), \quad w \in \{1, 2, \ldots, 2m-1\} \quad (6)$$

where $R_{w-m}(\vec{x}, \vec{y})$ is computed, in turn, as:

$$R_k(\vec{x}, \vec{y}) = \begin{cases} \sum_{l=1}^{m-k} x_{l+k} \cdot y_l, & k \geq 0 \\ R_{-k}(\vec{y}, \vec{x}), & k < 0 \end{cases} \quad (7)$$

Our goal is to compute the position w at which $CC_w(\vec{x}, \vec{y})$ is maximized. Based on this value of w, the optimal shift of $\vec{x}$ with respect to $\vec{y}$ is then $\vec{x}_{(s)}$, where $s = w - m$.

Depending on the domain or the application, different normalizations for $CC_w(\vec{x}, \vec{y})$ might be required. The most common normalizations are the biased estimator, NCC_b, the unbiased estimator, NCC_u, and the coefficient normalization, NCC_c, which are defined as follows:

$$NCC_q(\vec{x}, \vec{y}) = \begin{cases} \frac{CC_w(\vec{x}, \vec{y})}{m}, & q = \text{“}b\text{”}\ (NCC_b) \\ \frac{CC_w(\vec{x}, \vec{y})}{m - |w|}, & q = \text{“}u\text{”}\ (NCC_u) \\ \frac{CC_w(\vec{x}, \vec{y})}{\sqrt{R_0(\vec{x}, \vec{x}) \cdot R_0(\vec{y}, \vec{y})}}, & q = \text{“}c\text{”}\ (NCC_c) \end{cases} \quad (8)$$

Beyond the cross-correlation normalizations, time series might also require normalization to remove inherent distortions. Figure 3 illustrates how the cross-correlation normalizations for two sequences $\vec{x}$ and $\vec{y}$ of length $m = 1024$ are affected by time-series normalizations. (Appendix A also elaborates on the classification accuracy of cross-correlation variants under other time-series normalizations.) Independently of the normalization applied to $CC_w(\vec{x}, \vec{y})$, the produced sequence will have length 2047. Initially, in Figure 3a, we remove differences in amplitude by z-normalizing $\vec{x}$ and $\vec{y}$ in order to show that they are aligned and, hence, no shifting is required. If $CC_w(\vec{x}, \vec{y})$ is maximized for $w \in [1025, 2047]$ (or $w \in [1, 1023]$), one of $\vec{x}$ or $\vec{y}$ should be shifted by $i - 1024$ to the right (or $1024 - i$ to the left). Otherwise, if $w = 1024$, $\vec{x}$ and $\vec{y}$ are properly aligned, which is what we expect in our example. Figure 3b shows that if we do not z-normalize $\vec{x}$ and $\vec{y}$, and we use the biased estimator, then NCC_b is maximized at $w = 1797$, which indicates a shifting of a sequence to the left $1797 - 1024 = 773$ times. If we z-normalize $\vec{x}$ and $\vec{y}$, and use the unbiased estimator, then NCC_u is maximized at $w = 1694$, which indicates a shifting of a sequence to the right $1694 - 1024 = 670$ times (Figure 3c). Finally, if we z-normalize $\vec{x}$ and $\vec{y}$, and use the coefficient normalization, then NCC_c is maximized at $w = 1024$, which indicates that no shifting is required (Figure 3d).

As illustrated by the example above, normalizations of the data and the cross-correlation measure can have a sig-

Algorithm 1: $[dist, y'] = SBD(x, y)$

Input: Two z-normalized sequences x and y
Output: Dissimilarity $dist$ of x and y
Aligned sequence y' of y towards x

$length = 2^{nextpower2(2*length(x)-1)}$
$CC = IFFT\{FFT(x, length) * FFT(y, length)\}$ // Equation 12
$NCC_c = \frac{CC}{||x|| \, ||y||}$ // Equation 8
$[value, index] = max(NCC_c)$
$dist = 1 - value$ // Equation 9
$shift = index - length(x)$
if $shift \geq 0$ **then**
  $y' = [zeros(1, shift), y(1 : end - shift)]$ // Equation 5
else
  $y' = [y(1 - shift : end), zeros(1, -shift)]$ // Equation 5

nificant impact on the cross-correlation sequence produced, which makes the creation of a distance measure a non-trivial task. Furthermore, as we have seen in Figure 3, cross-correlation sequences produced by pairwise comparisons of multiple time series will differ in amplitude based on the normalizations. Thus, a normalization that produces values within a specified range should be used in order to meaningfully compare such sequences.

Shape-based distance (SBD): To devise a shape-based distance measure, and based on the previous discussion, we use the coefficient normalization that gives values between -1 and 1, regardless of the data normalization. Coefficient normalization divides the cross-correlation sequence by the geometric mean of autocorrelations of the individual sequences. After normalization of the sequence, we detect the position w where $NCC_c(\vec{x}, \vec{y})$ is maximized and we derive the following distance measure:

$$SBD(\vec{x}, \vec{y}) = 1 - \max_w \left(\frac{CC_w(\vec{x}, \vec{y})}{\sqrt{R_0(\vec{x}, \vec{x}) \cdot R_0(\vec{y}, \vec{y})}} \right) \quad (9)$$

which takes values between 0 to 2, with 0 indicating perfect similarity for time-series sequences.

Up to now we have addressed shift invariance. For scaling invariance, we transform each sequence $\vec{x}$ into $\vec{x}' = \frac{\vec{x} - \mu}{\sigma}$, so that its mean μ is zero and its standard deviation σ is one.

Efficient computation of SBD: From Equation 6, the computation of $CC_w(\vec{x}, \vec{y})$ for all values of w requires $\mathcal{O}(m^2)$ time, where m is the time-series length. The convolution theorem [39] states that the convolution of two time series can be computed as the Inverse Discrete Fourier Transform (IDFT) of the product of the individual Discrete Fourier Transforms (DFT) of the time series, where DFT is:

$$\mathcal{F}(x_k) = \sum_{r=0}^{|\vec{x}|-1} x_r e^{\frac{-2jrk\pi}{|\vec{x}|}}, \quad k = 0, \ldots, |\vec{x}| - 1 \quad (10)$$

and IDFT is:

$$\mathcal{F}^{-1}(x_r) = \frac{1}{|\vec{x}|} \sum_{k=0}^{|\vec{x}|-1} \mathcal{F}(x_k) e^{\frac{2jrk\pi}{|\vec{x}|}}, \quad r = 0, \ldots, |\vec{x}| - 1 \quad (11)$$

where $j = \sqrt{-1}$. Cross-correlation is then computed as the convolution of two time series if one sequence is first reversed in time, $\vec{x}^{(t)} = \vec{x}^{(-t)}$ [39], which equals taking the complex conjugate (represented by $*$) in the frequency domain. Thus, Equation 6 can be computed for every m as:

$$CC(\vec{x}, \vec{y}) = \mathcal{F}^{-1}\{\mathcal{F}(\vec{x}) * \mathcal{F}(\vec{y})\} \quad (12)$$

However, DFT and IDFT still require $\mathcal{O}(m^2)$ time. By using a Fast Fourier Transform (FFT) algorithm [15], the time reduces to $\mathcal{O}(m \log(m))$. Data and cross-correlation normalizations can also be efficiently computed; thus the overall time complexity of SBD remains $\mathcal{O}(m \log(m))$. Moreover, recursive algorithms compute an FFT by dividing it into pieces of power-of-two size [24]. Therefore, to further improve the performance of the FFT computation, when $CC(\vec{x}, \vec{y})$ is not an exact power of two we pad $\vec{x}$ and $\vec{y}$ with zeros to reach the next power-of-two length after $2m - 1$. Algorithm 1 shows how this efficient and parameter-free measure is computed in a few lines of code using modern mathematical software.

In this section, we showed effective cross-correlation and data normalizations to derive a shape-based distance measure. Importantly, we also discussed how the cross-correlation distance measure can be efficiently computed. In our experimental evaluation (Sections 4 and 5), we will show that SBD is highly competitive, achieving similar results to cDTW and DTW while being orders of magnitude faster. We now turn to the critical problem of extracting a centroid for a cluster, to represent the cluster data consistently with the shape-based distance measure described above.

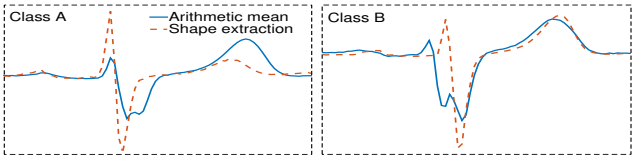


Figure 4: Examples of centroids for each class of the ECGFiveDays dataset, based on the arithmetic mean property (solid lines) and our shape extraction method (dashed lines).

3.2 Time-Series Shape Extraction

Many tasks in time-series analysis rely on methods that effectively summarize a set of time series by only one sequence. This summary sequence is often referred to as an *average sequence* or, in the context of clustering, as a *centroid*. The extraction of meaningful centroids is a challenging task that critically depends on the choice of distance measure (see Section 2.1). We now show how to determine such centroids for time-series clustering for the SBD distance measure, to capture shared characteristics of the underlying data.

The easiest way to extract an average sequence from a set of sequences is to compute each coordinate of the average sequence as the arithmetic mean of the corresponding coordinates of all sequences. This approach is used by k-means, the most popular clustering method. In Figure 4, the solid lines show such centroids for each class in the ECGFiveDays dataset of Figure 1: these centroids do not capture effectively the class characteristics (see Figures 1 and 4).

To avoid such problems, we cast the centroid computation as an optimization problem where the objective is to find the minimizer of the sum of squared distances to all other time series sequences (Equation 2). However, as cross-correlation intuitively captures the similarity — rather than the dissimilarity — of time series, we can express the computed sequence as the maximizer $\mu_k^\star$ of the squared similarities to all other time-series sequences. By rewriting Equation 2 as a maximization problem and from Equation 8, we obtain:

$$\vec{\mu_k}^\star = \operatorname*{argmax}_{\vec{\mu_k}} \sum_{\vec{x_i} \in P_k} NCC_c(\vec{x_i}, \vec{\mu_k})^2$$

$$= \operatorname*{argmax}_{\vec{\mu_k}} \sum_{\vec{x_i} \in P_k} \left(\frac{\max_w CC_w(\vec{x_i}, \vec{\mu_k})}{\sqrt{R_0(\vec{x_i}, \vec{x_i}) \cdot R_0(\vec{\mu_k}, \vec{\mu_k})}} \right)^2 \quad (13)$$

Equation 13 requires the computation of an optimal shift for every $\vec{x_i} \in P_k$. As this approach is used in the context of it-

Algorithm 2: $C' = ShapeExtraction(X, C)$

Input: X is an n-by-m matrix with z-normalized time series.
C is a 1-by-m vector with the reference sequence against which time series of X are aligned.
Output: C' is a 1-by-m vector with the centroid.

```
1  X' ← [ ]
2  for i ← 1 to n do
3  |   [dist, x'] ← SBD(C, X(i))                    // Algorithm 1
4  |   X' ← [X'; x']
5  S ← X'^T · X'                                    // S of Equation 15
6  Q ← I − (1/m) · O                                // Q of Equation 15
7  M ← Q^T · S · Q                                  // M of Equation 15
8  C' ← Eig(M, 1)                                   // Extract first eigenvector
```

erative clustering, we use the previously computed centroid as reference and align all sequences towards this reference sequence. This is a reasonable choice because the previous centroid will be very close to the new centroid. For this alignment, we use SBD, which identifies an optimal shift for every $\vec{x_i} \in P_k$. Subsequently, as sequences are already aligned towards a reference sequence before the centroid computation, we can also omit the denominator of Equation 13. Then, by combining Equations 6 and 7, we obtain:

$$\vec{\mu_k}^{\star} = \underset{\vec{\mu_k}}{\operatorname{argmax}} \sum_{\vec{x_i} \in P_k} \left(\sum_{l \in [1,m]} x_{il} \cdot \mu_{kl} \right)^2$$

For simplicity, we express this equation with vectors and assume that the $\vec{x_i}$ sequences have already been z-normalized to handle the differences in amplitude:

$$\begin{aligned} \vec{\mu_k}^{\star} &= \underset{\vec{\mu_k}}{\operatorname{argmax}} \sum_{\vec{x_i} \in P_k} (\vec{x_i}^T \cdot \vec{\mu_k})^2 \\ &= \underset{\vec{\mu_k}}{\operatorname{argmax}} \vec{\mu_k}^T \cdot \sum_{\vec{x_i} \in P_k} (\vec{x_i} \cdot \vec{x_i}^T) \cdot \vec{\mu_k} \end{aligned} \tag{14}$$

In the previous equation, only $\vec{\mu_k}$ is not z-normalized. To handle the centering (i.e., the subtraction of mean) of $\vec{\mu_k}$ we set $\vec{\mu_k} = \vec{\mu_k} \cdot Q$, where $Q = I - \frac{1}{m} O$, I is the identity matrix, and O is a matrix with all ones. Furthermore, to make $\vec{\mu_k}$ have unit norm, we divide Equation 14 by $\vec{\mu_k}^T \cdot \vec{\mu_k}$. Finally, by substituting S for $\sum_{\vec{x_i} \in P_k} (\vec{x_i} \cdot \vec{x_i}^T)$, we obtain:

$$\begin{aligned} \vec{\mu_k}^{\star} &= \underset{\vec{\mu_k}}{\operatorname{argmax}} \frac{\vec{\mu_k}^T \cdot Q^T \cdot S \cdot Q \cdot \vec{\mu_k}}{\vec{\mu_k}^T \cdot \vec{\mu_k}} \\ &= \underset{\vec{\mu_k}}{\operatorname{argmax}} \frac{\vec{\mu_k}^T \cdot M \cdot \vec{\mu_k}}{\vec{\mu_k}^T \cdot \vec{\mu_k}} \end{aligned} \tag{15}$$

where $M = Q^T \cdot S \cdot Q$. Through the above transformations, we have reduced the optimization of Equation 13 to the optimization of Equation 15, which is a well-known problem called maximization of the Rayleigh Quotient [30]. We can find the maximizer $\vec{\mu_k}^{\star}$ as the eigenvector that corresponds to the largest eigenvalue of the real symmetric matrix M.

Algorithm 2 shows how we can extract the most representative shape from the underlying data in a few lines of code. In Figure 4, the dashed lines show the centroids of each class in the ECGFiveDays dataset, extracted with Algorithm 2 and using randomly selected sequences as reference sequences. This method for shape extraction can more effectively capture the characteristics of each class (Figure 1) than by using the arithmetic mean property (solid lines in Figure 4). In the next section, we show how our shape extraction method is used in a time-series clustering algorithm.

Algorithm 3: $[IDX, C] = k\text{-}Shape(X, k)$

Input: X is an n-by-m matrix containing n time series of length m that are initially z-normalized.
k is the number of clusters to produce.
Output: IDX is an n-by-1 vector containing the assignment of n time series to k clusters (initialized randomly).
C is a k-by-m matrix containing k centroids of length m (initialized as vectors with all zeros).

```
1  iter ← 0
2  IDX' ← [ ]
3  while IDX! = IDX' and iter < 100 do
4  |   IDX' ← IDX
   |   // Refinement step
5  |   for j ← 1 to k do
6  |   |   X' ← [ ]
7  |   |   for i ← 1 to n do
8  |   |   |   if IDX(i) = j then
9  |   |   |   |   X' ← [X'; X(i)]
10 |   |   C(j) ← ShapeExtraction(X', C(j))          // Algorithm 2
   |   // Assignment step
11 |   for i ← 1 to n do
12 |   |   mindist ← ∞
13 |   |   for j ← 1 to k do
14 |   |   |   [dist, x'] ← SBD(C(j), X(i))          // Algorithm 1
15 |   |   |   if dist < mindist then
16 |   |   |   |   mindist ← dist
17 |   |   |   |   IDX(i) ← j
18 |   iter ← iter + 1
```

3.3 Shape-based Time-Series Clustering

We now present k-Shape, our novel algorithm for time-series clustering. k-Shape relies on the SBD distance measure of Section 3.1 and the shape extraction method of Section 3.2 to efficiently produce clusters of time series.

k-Shape Clustering Algorithm: k-Shape is a partitional clustering method that is based on an iterative refinement procedure similar to the one used in k-means. Through this iterative procedure, k-Shape minimizes the sum of squared distances (Equation 1) and manages to: (i) produce homogeneous and well-separated clusters, and (ii) scale linearly with the number of time series. Our algorithm compares sequences efficiently and computes centroids effectively under the scaling-, translation-, and shift invariances. k-Shape is a non-trivial instantiation of k-means and, in contrast to similar attempts in the literature [64, 87], its distance measure and centroid computation method make k-Shape the only scalable method that significantly outperforms k-means.

In every iteration, k-Shape performs two steps: (i) in the assignment step, the algorithm updates the cluster memberships by comparing each time series with all computed centroids and by assigning each time series to the cluster of the closest centroid; (ii) in the refinement step, the cluster centroids are updated to reflect the changes in cluster memberships in the previous step. The algorithm repeats these two steps until either no change in cluster membership occurs or the maximum number of iterations allowed is reached. In the assignment step, k-Shape relies on the distance measure of Section 3.1, whereas in the refinement step it relies on the centroid computation method of Section 3.2.

k-Shape (see Algorithm 3) expects as input the time series set X and the number of clusters k that we want to produce. Initially, we randomly assign the time series in X to clusters. Then, we compute each cluster centroid using Algorithm 2 (lines 5-10). Once the centroids are computed, we refine the memberships of the clusters by using the SBD distance measure (Algorithm 1) in lines 11-17. We repeat this procedure until the algorithm converges or reaches the maximum

number of iterations (usually a small number, such as 100). The output of the algorithm is the assignment of sequences to clusters and the centroids for each cluster.

Complexity of k-Shape: As we claimed earlier, k-Shape scales linearly with the number of time series. To see why, we will analyze the computational complexity of Algorithm 3, where n is the number of time series, k is the number of clusters, and m is the length of the time series. In the assignment step, k-Shape computes the dissimilarity of n time series to k centroids by using SBD, which requires $\mathcal{O}(m \cdot \log(m))$ time. Thus, the time complexity of this step is $\mathcal{O}(n \cdot k \cdot m \cdot \log(m))$. In the refinement step, for every cluster, k-Shape computes matrix M, which requires $\mathcal{O}(m^2)$ time, and performs an eigenvalue decomposition on M, which requires $\mathcal{O}(m^3)$ time. Thus, the complexity of this step is $\mathcal{O}(\max\{n \cdot m^2, k \cdot m^3\})$. Overall, k-Shape requires $\mathcal{O}(\max\{n \cdot k \cdot m \cdot \log(m), n \cdot m^2, k \cdot m^3\})$ time per iteration. We see that this algorithm has a linear dependence in the number of time series, and the majority of the computation cost depends on the length of the time series. However, this length is usually much smaller than the number of time series (i.e., $m \ll n$) and, hence, the dependence on m is not a bottleneck. (Appendix B further examines the scalability of k-Shape.) In rare cases where m is very large (i.e., $m \gg n$), segmentation or dimensionality reduction approaches can be used to sufficiently reduce the length of the sequences [10, 49].

We now turn to the experimental evaluation of k-Shape against the state-of-the-art time-series clustering approaches.

4. EXPERIMENTAL SETTINGS

In this section, we describe the experimental settings for the evaluation of both SBD and our k-Shape algorithm.

Datasets: We use the largest public collection of class-labeled time-series datasets, namely, the UCR time-series collection [1]. It consists of 48 datasets,[4] both synthetic and real, which span several different domains. Each dataset contains from 56 to 9236 sequences. The sequences in each dataset have equal length, but from one dataset to another the sequence length varies from 24 to 1882. These datasets are annotated and every sequence can belong to only one class, which, in the context of clustering, should be interpreted as the cluster where the sequence belongs. Furthermore, the datasets are already z-normalized[5] and split into training and test sets. As we will see, we use this split of the datasets for the distance measure evaluation; we also use the training sets for tuning some of the baselines.

Platform: We ran our experiments on a cluster of 10 servers with identical configuration: Dual Intel Xeon X5550 (4-core with 2-way SMT) processor with clock speed at 2.67 GHz and 24 GB RAM. We utilized up to 16 processes continuously for a period of two months in order to perform all experiments included in this paper. Each server runs Ubuntu 12.04 (64-bit) and Matlab R2012b (64-bit).

Implementation: We implemented our approach and all state-of-the-art approaches that we compare against under the same framework, in Matlab, for a consistent evaluation in terms of both accuracy and efficiency. For repeatability purposes, we make all datasets and source code available.[6]

Baselines: We evaluate both our distance measure, SBD, and our clustering approach, k-Shape. We compare SBD against the strongest state-of-the-art distance measures for time series (see Section 2.3 for a detailed discussion):

- **ED:** a simple, efficient — yet accurate — distance measure for time series [20]
- **DTW:** the best performing — but expensive — distance measure for time series [72]
- **cDTW:** the constrained version of DTW, with improved accuracy and efficiency [72]

We compare k-Shape against the three strongest types of scalable and non-scalable clustering methods, namely, partitional, hierarchical, and spectral methods (see Section 2.4 for a detailed discussion), combined with the most competitive distance measures. As scalable methods, we consider the classic k-means algorithm with ED (k-AVG+ED) [50] and the following variants of k-means:

- **k-AVG+Dist:** k-means with DTW and SBD as distance measures and the arithmetic mean of time series coordinates for centroid computation
- **k-DBA:** k-means with DTW as distance measure and the DBA method for centroid computation [64]
- **KSC:** k-means with a distance measure offering pairwise scaling and shifting of time series and computation of the spectral norm of a matrix (i.e., matrix decomposition) for centroid computation [87]

As non-scalable methods, among partitional methods we consider the Partitioning Around Medoids (PAM) implementation of the k-medoids algorithm [40]. Among hierarchical methods, we use agglomerative hierarchical clustering with single, average, and complete linkage criteria [40]. Finally, among spectral methods, we consider the popular normalized spectral clustering method [58]. These non-scalable approaches require a large number of distance calculations to compute the full dissimilarity matrix and, hence, they become unacceptably inefficient with high-cost distance measures. For this reason, we ignore DTW and focus instead on ED, cDTW, and SBD as distance measures. In Table 1 we summarize the combinations used in our evaluation. Overall, we compared k-Shape against 20 clustering approaches.

Parameter settings: Among the distance measures discussed above, only cDTW requires setting a parameter, to constrain its warping window. We consider two cases from the literature: (i) cDTW^{opt}: we compute the optimal window by performing a leave-one-out classification step over the training set of each dataset; (ii) cDTW^{w}: we use as window 5%, for cDTW^5, and 10%, for cDTW^{10}, of the length of the time series of each dataset; this second case is more realistic for an unsupervised setting such as clustering.[7] For the 1-NN classification computation, we also consider the state-of-the-art lower bounding approach LB$_{Keogh}$ [44], which prunes time series unlikely for a match when DTW and cDTW are used. We denote lower bounding with the LB subscript (e.g., cDTW$^{10}_{LB}$). For clustering, all the algorithms that we compare, except for hierarchical clustering, receive the target number of clusters k as input, which is equal to the number of classes for each dataset. For hierarchical clustering, we compute a threshold that cuts the produced dendrogram at the minimum height such that k clusters are formed. We set the maximum number of iterations for k-Shape, all variants of k-means, PAM, and spectral methods to 100. Finally, in every iteration of k-Shape, k-DBA, and KSC, we use the centroids of the previous run

[4] Currently, 47 out of 48 datasets are listed in [1]. One dataset, namely, Insect, will become publicly available in the next release, according to the owners of the UCR repository [1].

[5] Several datasets as available in [1] are not z-normalized. (We have communicated about this issue with the site owners.) Therefore, our experiments start with a z-normalization step for all datasets.

[6] http://www.cs.columbia.edu/~jopa/kshape.html

[7] For non-scalable methods, we use cDTW^5 for its efficiency, as noted in Table 1, even though cDTW^{10} and cDTW^{opt} perform similarly.

Name	Clustering Algorithm	Distance Measure
PAM+ED	Partitioning Around Medoids	ED
PAM+cDTW	Partitioning Around Medoids	cDTW^5
PAM+SBD	Partitioning Around Medoids	SBD
H-S+ED	Hierarchical with *single* linkage	ED
H-A+ED	Hierarchical with *average* linkage	ED
H-C+ED	Hierarchical with *complete* linkage	ED
H-S+cDTW	Hierarchical with *single* linkage	cDTW^5
H-A+cDTW	Hierarchical with *average* linkage	cDTW^5
H-C+cDTW	Hierarchical with *complete* linkage	cDTW^5
H-S+SBD	Hierarchical with *single* linkage	SBD
H-A+SBD	Hierarchical with *average* linkage	SBD
H-C+SBD	Hierarchical with *complete* linkage	SBD
S+ED	Normalized Spectral Clustering	ED
S+cDTW	Normalized Spectral Clustering	cDTW^5
S+SBD	Normalized Spectral Clustering	SBD

Table 1: Combinations of PAM, hierarchical, and spectral methods with ED, cDTW, and SBD for our evaluation.

as reference sequences to refine the centroids of the current run once.

Metrics: We compare all approaches on both runtime and accuracy. For runtime, we compute CPU time utilization and report time ratios for our comparisons. Following [19], we use the 1-NN classifier, which is a simple and parameter-free classifier, to evaluate distance measures. We report the classification accuracy (i.e., number of correctly classified instances over all instances) by performing 1-NN classification over the training and test sets of each dataset. Because the 1-NN classifier is deterministic, we make this computation once. Following [7], we use the Rand Index [67] to evaluate clustering accuracy over the fused training and test sets of each dataset. This metric is related to the classification accuracy and is defined as $R = \frac{TP+TN}{TP+TN+FP+FN}$, where TP is the number of time series pairs that belong to the same class and are assigned to the same cluster, TN is the number of time series pairs that belong to different classes and are assigned to different clusters, FP is the number of time series pairs that belong to different classes but are assigned to the same cluster, and FN is the number of time series pairs that belong to the same class but are assigned to different clusters. As hierarchical algorithms are deterministic, we report the Rand Index over one run. However, for partitional methods, we report the average Rand Index over 10 runs and for spectral methods the average Rand Index over 100 runs; in every run we use a different random initialization.

Statistical analysis: Following [7, 26], we analyze the results of every pairwise comparison of algorithms over multiple datasets using the Wilcoxon test [84] with a 99% confidence level. According to [17], the Wilcoxon test is less affected by outliers than is the t-test [69], as Wilcoxon does not consider absolute commensurability of differences. Moreover, using pairwise tests to reason about multiple algorithms is not fully satisfactory because sometimes the null hypotheses are rejected due to random chance. Therefore, we also use the Friedman test [23] followed by the post-hoc Nemenyi test [57] for comparison of multiple algorithms over multiple datasets, as suggested in [17]. The Friedman and Nemenyi tests require more evidence to detect statistical significance than the Wilcoxon test [26] (i.e., the larger the number of methods, the larger the number of datasets required) and, hence, as we already use all 48 datasets for the Wilcoxon test, we report statistical significant results with a 95% confidence level.

5. EXPERIMENTAL RESULTS

In this section, we discuss our experiments. First, we evaluate SBD against the state-of-the-art distance measures (Section 5.1). Then, we compare k-Shape against scalable (Section 5.2) and non-scalable (Section 5.3) clustering approaches. Finally, we highlight our findings (Section 5.4).

5.1 Evaluation of SBD

Comparison against ED: To understand if SBD (Section 3.1) is an effective measure for time-series comparison, we evaluate it against the state-of-the-art distance measures, using their 1-NN classification accuracies across 48 datasets (Section 4). Table 2 reports the performance of the state-of-the-art measures against the baseline ED. All distance measures, including SBD, outperform ED with statistical significance. For DTW, constraining its warping window significantly improves performance. In particular, cDTW^{opt} performs at least as well as DTW in 36 datasets, cDTW^5 performs at least as well as DTW in 34 datasets, and cDTW^{10} performs at least as well as DTW in 41 datasets. However, the statistical test suggests that there is no significant difference between cDTW^{opt}, cDTW^5, and cDTW^{10}. Figure 5a further illustrates the superiority of SBD over ED.

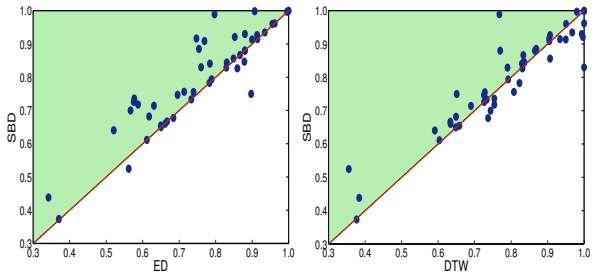
Comparison against DTW and cDTW: Figure 5b shows that the difference in accuracy between SBD and DTW is in most cases negligible: SBD performs at least as well as DTW in 30 datasets, but the statistical test reveals no evidence that either measure is better than the other. Considering the constrained versions of DTW, we observe that SBD performs similarly to or better than cDTW^{opt}, cDTW^5, and cDTW^{10} in 22, 18, and 19 datasets, respectively. Interestingly, there is no significant difference between SBD and cDTW^{10}, but cDTW^{opt} and cDTW^5 are significantly better than SBD. cDTW^{opt}, with its optimal tuning, identifies that across the 48 datasets the average warping window is 4.5%. This explains why cDTW^5 behaves similarly to cDTW^{opt} and outperforms SBD, whereas the large majority of constrained cDTW^w variants, including cDTW^{10}, do not outperform SBD. Importantly, the p-values for the Wilcoxon test between cDTW^{opt} and SBD, and between cDTW^5 and SBD, are close to the confidence level, which indicates that deeper statistical analysis is required, as we discuss next.

Statistical analysis: To better understand the performance of SBD in comparison with cDTW^{opt} and cDTW^5, we evaluate the significance of their differences in accuracy when considered all together. Figure 6 shows the average rank across datasets of each distance measure. cDTW^{opt} is the top measure, with an average rank of 1.96, meaning that cDTW^{opt} performed best in the majority of the datasets. The Friedman test rejects the null hypothesis that all measures behave similarly, and, hence, we proceed with a post-hoc Nemenyi test, to evaluate the significance of the differences in the ranks. The wiggly line in the figure connects all measures that do not perform statistically differently according to the Nemenyi test. We observe that the ranks of cDTW^{opt}, cDTW^5, and SBD do not present a significant difference, and ED, which is ranked last, is significantly worse than the others. In conclusion, SBD is a very competitive distance measure that significantly outperforms ED and achieves similar results to both constraint and unconstraint versions of DTW. Moreover, SBD is the most robust variant of the cross-correlation measure (see Appendix A).

Efficiency: We now show that SBD is not only competitive in terms of accuracy, but also highly efficient. We also

Distance Measure	>	=	<	Better	Average Accuracy	Runtime
DTW **DTW$_{LB}$**	29	2	17	✓	0.788	15573x 6040x
cDTW^{opt} **cDTW$^{opt}_{LB}$**	31	15	2	✓	0.814	2873x 322x
cDTW^{5} **cDTW$^{5}_{LB}$**	34	3	11	✓	0.809	1558x 122x
cDTW^{10} **cDTW$^{10}_{LB}$**	33	1	14	✓	0.804	2940x 364x
SBD$_{NoFFT}$ **SBD$_{NoPow2}$** **SBD**	30	12	6	✓	0.795	224x 8.7x 4.4x

Table 2: Comparison of distance measures. Columns ">", "=", and "<" denote the number of datasets over which a distance measure is better, equal, or worse, respectively, in comparison to ED. "Better" indicates that a distance measure outperforms ED with statistical significance. "Average accuracy" denotes the accuracy achieved in the 48 datasets whereas "Runtime" indicates the factor by which a distance measure is slower than ED.



(a) SBD vs. ED (b) SBD vs. DTW

Figure 5: Comparison of SBD, ED, and DTW over 48 datasets. Circles above the diagonal indicate datasets over which SBD has better accuracy than the compared method.

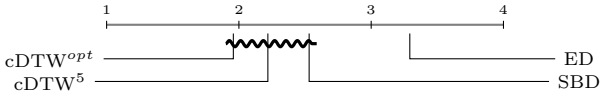


Figure 6: Ranking of distance measures based on the average of their ranks across datasets. The wiggly line connects all measures that do not perform statistically differently according to the Nemenyi test.

demonstrate that implementation choices for SBD can significantly impact its speed. The last row of Table 2 shows the factors by which each SBD variation is slower than ED. The optimized version of SBD, denoted simply as SBD, is the fastest version, performing only 4.4x slower than ED. When we use SBD still with FFT but without the power-of-two-length optimization discussed in Section 3.1, the resulting measure, SBD$_{NoPow2}$, is 8.7x slower than ED. Furthermore, SBD$_{NoFFT}$, the version of SBD without FFT, is two orders of magnitude slower (224x) than ED and one order slower (51x) than SBD. Table 2 also shows that SBD is substantially faster than the DTW and cDTW variants. Specifically, SBD is three orders of magnitude (3533x) faster than DTW, two orders (652x) faster than cDTW^{opt}, two orders (353x) faster than cDTW^{5}, and two orders (667x) faster than cDTW^{10}. Even when we speed up the search of 1-NN classification computation, by pruning time series unlikely for a match using LB$_{Keogh}$, SBD is still significantly faster. In particular, SBD now becomes one order of magnitude faster (73x) than cDTW$^{opt}_{LB}$, one order faster (28x) than cDTW$^{5}_{LB}$, and one order faster (82.5x) than cDTW$^{10}_{LB}$, while remaining three orders faster (1370x) than DTW$_{LB}$.

5.2 k-Shape Against Other Scalable Methods

Comparison against k-AVG+ED: Having shown the robustness of SBD, we now compare our algorithm, k-Shape, against scalable state-of-the-art time-series clustering algorithms. Table 3 reports the performance of variants of k-means against k-AVG+ED, using their Rand Index on the 48 datasets (see Section 4). From all these variants of k-means, only k-Shape outperforms k-AVG+ED with statistical significance, and, in particular, k-Shape significantly outperforms every other method. In most cases, replacing ED in k-means with other distance measures not only does not improve accuracy significantly, but in certain cases results in substantially lower performance. For example, k-AVG+SBD achieves higher accuracy than k-AVG+ED in 67% of the datasets, but the differences in accuracy are not statistically significant. Interestingly, when DTW is combined with k-means, in k-AVG+DTW, the performance is significantly worse than with k-AVG+ED. Even in cases where both the distance measure and the centroid computation method of k-means are modified, the performance does not improve significantly in comparison to k-AVG+ED. k-DBA outperforms k-AVG+DTW with statistical significance in 39 out of 48 datasets. Both of these approaches use DTW as distance measure, but k-DBA also modifies its centroid computation method. This modification significantly improves the performance of k-DBA over that of k-AVG+DTW, with an average improvement in Rand Index of 25.6%. Despite this improvement, k-DBA still does not perform better than k-AVG+ED in a statistical significant manner.[8] Another algorithm that modifies both the distance measure and the centroid computation method of k-means is KSC. Similarly to k-DBA, KSC does not outperform k-AVG+ED.

Comparison against k-DBA and KSC: Both k-DBA and KSC are similar to k-Shape in that they all modify both the centroid computation method and the distance measure of k-means. Therefore, to understand the impact of these modifications, we compare them against our k-Shape algorithm (see Figure 7). k-Shape is better in 30 datasets and worse in 18 datasets in comparison to KSC (Figure 7a), and is better in 35 datasets and worse in 13 datasets in comparison to k-DBA (Figure 7b). In both of these comparisons, the statistical test indicates the superiority of k-Shape.

Statistical analysis: In addition to the pairwise comparisons performed with the Wilcoxon test, we further evaluate the significance of the differences in algorithm performance when considered all together. Figure 8 shows the average rank across datasets of each k-means variant. k-Shape is the top technique, with an average rank of 1.89, meaning that k-Shape achieved better rank in the majority of the datasets. The Friedman test rejects the null hypothesis that all algorithms behave similarly, and, hence, we proceed with a post-hoc Nemenyi test, to evaluate the significance of the differences in the ranks. We observe that the ranks of KSC, k-DBA, and k-AVG+ED do not present a statistically significant difference, whereas k-Shape, which is ranked first, is significantly better than the others. To conclude, modi-

[8] Even when multiple refinements of k-DBA's centroids are performed per iteration, k-DBA still does not outperform k-AVG+ED. In particular, performing five refinements per iteration improves the Rand Index by 4% but runtime increases by 30%.

Algorithm	>	=	<	Better	Worse	Rand Index	Runtime
k-AVG+SBD	32	1	15	✗	✗	0.745	3.6x
k-AVG+DTW	10	0	38	✗	✓	0.584	3444x
KSC	22	0	26	✗	✗	0.636	448x
k-DBA	18	0	30	✗	✗	0.733	3892x
k-Shape+DTW	19	1	28	✗	✗	0.698	4175x
k-Shape	36	1	11	✓	✗	0.772	12.4x

Table 3: Comparison of k-means variants against k-AVG+ED. Columns ">", "=", and "<" denote the number of datasets over which an algorithm is better, equal, or worse, respectively, in comparison to k-AVG+ED. "Better" indicates that an algorithm outperforms k-AVG+ED with statistical significance whereas "Worse" indicates that k-AVG+ED outperforms an algorithm with statistical significance. "Rand Index" denotes the accuracy achieved in the 48 datasets whereas "Runtime" indicates the factor by which an algorithm is slower than k-AVG+ED.

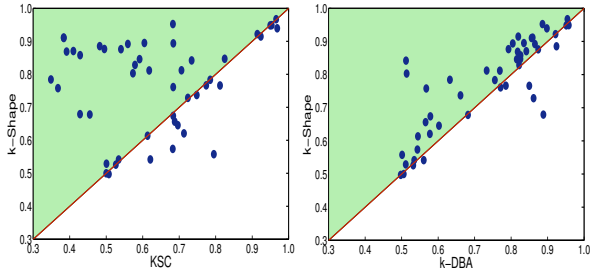
fying k-means with inappropriate distance measures or centroid computation methods might lead to unexpected results. The same holds for k-Shape, where the use of DTW as its distance measure, in k-Shape+DTW, significantly drops its performance (i.e., k-Shape outperforms k-Shape+DTW with statistical significance in 36 out of 48 datasets).

Efficiency: k-Shape is the only algorithm that outperforms all k-means variants, including the simple, yet robust, k-AVG+ED. We now investigate whether k-Shape's superiority in terms of accuracy has an associated penalty in efficiency. Table 3 shows the factors by which each algorithm is slower than k-AVG+ED. Our approach, k-Shape, is one order of magnitude faster (36x) than KSC, two orders of magnitude faster (313x) than k-DBA, and 12.4x slower than k-AVG+ED.

5.3 k-Shape Against Non-Scalable Methods

Comparison against k-AVG+ED: Up to now, we have focused our evaluation on scalable clustering algorithms. In order to show the robustness of k-Shape in terms of accuracy beyond scalable approaches, we now ignore scalability and compare k-Shape against clustering methods that scale quadratically with the number of time series, namely, hierarchical, spectral, and k-medoids methods. Table 4 reports the performance of non-scalable methods against k-AVG+ED. Among all existing state-of-the-art methods that use ED or cDTW as distance measures, only partitional methods perform similarly to or better than k-AVG+ED. In particular, PAM+cDTW is the only method that outperforms k-AVG+ED with statistical significance. PAM+ED achieves better performance in 63% of the datasets in comparison to k-AVG+ED; however, this difference is not statistically significant. Moreover, PAM+cDTW performs better in 33 datasets, equal in 1 dataset, and worse in 14 datasets relative to PAM+ED. For this comparison, the statistical significance test indicates the superiority of PAM+cDTW.

All combinations of hierarchical clustering, with all different linkage criteria, perform poorly in comparison to k-AVG+ED. Interestingly, k-AVG+ED outperforms all of them with statistical significance. We observe that the major difference in performance among hierarchical methods is the linkage criterion and not the distance measure. This highlights the importance of the clustering method and not only of the distance measure for time-series clustering. Similarly to hierarchical methods, spectral methods also perform poorly against k-AVG+ED. S+cDTW performs better in more datasets than S+ED in comparison to k-AVG+ED:



(a) k-Shape vs. KSC (b) k-Shape vs. k-DBA

Figure 7: Comparison of k-Shape, KSC, and k-DBA over 48 datasets. Circles above the diagonal indicate datasets over which k-Shape has better Rand Index.

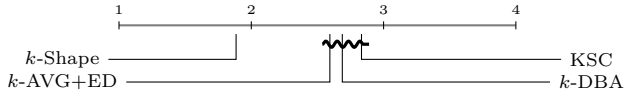


Figure 8: Ranking of k-means variants based on the average of their ranks across datasets. The wiggly line connects all techniques that do not perform statistically differently according to the Nemenyi test.

comparing S+cDTW with S+ED, S+cDTW achieves similar or better accuracy in 27 datasets, but this difference is not significant. Importantly, k-AVG+ED is significantly better than both S+ED and S+cDTW. Therefore, for hierarchical and spectral methods, these distance measures have a small impact on their performance.

k-Shape against PAM+cDTW: Among all methods that we evaluated, only PAM+cDTW outperforms k-AVG+ED with statistical significance. Therefore, we compare this approach with k-Shape. PAM+cDTW is better in 31, equal in 1, and worse in 16 datasets in comparison to k-Shape, but this difference is not significant. For completeness, we also evaluate SBD with hierarchical, spectral, and k-medoids methods. For hierarchical methods, H-C+SBD is better than H-A+SBD, and, in turn, H-A+SBD is better than H-S+SBD, all with statistical significance. For each linkage option, there is no significance in the accuracy between ED, SBD, and cDTW. S+SBD, in contrast to S+ED and S+cDTW, outperforms k-AVG+ED in 38 out of 48 datasets, with statistical significance. S+SBD is also significantly better than S+ED and S+cDTW, but S+SBD does not outperform k-Shape. Similarly, PAM+SBD performs equally to or better than k-AVG+ED in 36 out of 48 datasets. The statistical test suggests that PAM+SBD is significantly better than k-AVG+ED but not better than k-Shape.

Statistical analysis: We evaluate the significance of the differences in algorithm performance for all algorithms that significantly outperform k-AVG+ED. The Friedman test rejects the null hypothesis that all algorithms behave similarly, and, hence, as before, we proceed with a post-hoc Nemenyi test. Figure 9 shows that k-Shape, PAM+SBD, PAM+cDTW, and S+SBD do not present a significant difference in accuracy, whereas k-AVG+ED, which is ranked last, is significantly worse than the others.

From our extensive evaluation of existing scalable and non-scalable clustering approaches for time series that use ED, cDTW, or DTW as distance measures, PAM+cDTW is the only approach that achieves similar — but not better — results to k-Shape. In contrast to k-Shape, PAM+cDTW has two drawbacks that make it an unrealistic choice for time-series clustering: (i) its distance measure, cDTW, re-

Algorithm	>	=	<	Better	Worse	Rand Index
H-S+ED	3	1	44	✗	✓	0.328
H-S+cDTW	7	0	41	✗	✓	0.371
H-S+SBD	6	1	41	✗	✓	0.349
H-A+ED	3	1	44	✗	✓	0.599
H-A+cDTW	9	0	39	✗	✓	0.617
H-A+SBD	8	0	40	✗	✓	0.541
H-C+ED	8	0	40	✗	✓	0.690
H-C+cDTW	15	0	33	✗	✓	0.699
H-C+SBD	17	0	31	✗	✓	0.697
S+ED	7	1	40	✗	✓	0.602
S+cDTW	18	1	29	✗	✓	0.563
S+SBD	38	0	10	✓	✗	0.769
PAM+ED	30	1	17	✗	✗	0.762
PAM+cDTW	38	1	9	✓	✗	0.772
PAM+SBD	35	1	12	✓	✗	0.780

Table 4: Comparison of hierarchical, spectral, and k-medoids variants against k-AVG+ED. Columns ">", "=", and "<" denote the number of datasets over which an algorithm is better, equal, or worse, respectively, in comparison to k-AVG+ED. "Better" indicates that an algorithm outperforms k-AVG+ED with statistical significance whereas "Worse" indicates that k-AVG+ED outperforms an algorithm with statistical significance. "Rand Index" denotes the accuracy achieved in the 48 datasets.

quires tuning to improve its performance and reduce the computation cost; and (ii) the computation of the dissimilarity matrix that PAM+cDTW requires as input makes it unable to scale in both time and space. For example, the matrix computation alone is already two orders of magnitude slower than the computation required by k-Shape. Thus, k-Shape emerges as a domain-independent, highly accurate, and scalable approach for time-series clustering.

5.4 Summary of Results

In short, our experimental evaluation suggests that: (1) cross-correlation measures (e.g., SBD), which are not widely adopted as time-series distance measures, are as competitive as state-of-the-art measures, such as cDTW and DTW, but significantly faster; (2) the k-means algorithm with ED, in contrast to what has been reported in the literature, is a robust approach for time-series clustering, but inadequate modifications of its distance measure and centroid computation can reduce its performance; (3) the choice of clustering method, which was believed to be less important than that of distance measure, is as important as the choice of distance measure; and overall (4) k-Shape is a highly accurate and efficient method for time-series clustering.

6. RELATED WORK

This paper focused on efficient and domain-independent time-series clustering. Section 2 provided an in-depth discussion of the state of the art for time-series clustering, which we will not repeat for brevity. Specifically, Section 2.1 summarized the relevant theoretical background, Section 2.2 reviewed common distortions in time series, and Section 2.3 discussed the most popular state-of-the-art distance measures for time series. (We refer the reader to [19, 81] for a thorough review and evaluation of the alternative time-series distance measures.) Section 2.4 highlighted existing approaches for clustering time series. (We refer the reader to [83] for a more detailed view of these approaches.) Finally, Section 2.5 discussed the methods for centroid computation that are part of many time-series clustering algorithms.

As argued throughout the paper, we focus on shape-based clustering of time series. Beyond shape-based clustering

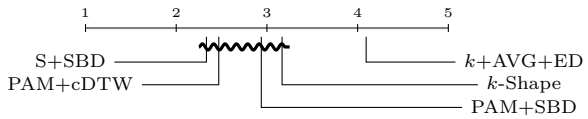


Figure 9: Ranking of methods that outperform k-AVG+ED based on the average of their ranks across datasets. The wiggly line connects all techniques that do not perform statistically differently according to the Nemenyi test.

algorithms, statistical-based approaches use measures that summarize characteristics [82], coefficients of models [38], or portions of time series (i.e., shapelets) [89] as descriptive features to cluster time series. Unfortunately, statistical-based approaches frequently require the non-trivial tuning of multiple parameters, which often leads to ad-hoc decisions, or their effectiveness has been established only for isolated domains and not in a domain-independent manner. Instead, shape-based approaches are general and leverage the vast literature on distance measures.

The best performing shape-based approaches from the literature are partitional methods combined with scale- and shift-invariant distance measures. Among partitional methods, k-medoids [40] is the most popular method as it enables the easy adoption of any shape-based distance measure [19, 81]. However, k-medoids requires the computation of the full dissimilarity matrix among all time series, which makes it particularly slow and unable to scale. Recent alternative approaches [32, 53, 59, 64, 87] have focused on k-means [50], which is scalable but requires the modification of the centroid computation method when the distance measure is altered, in order to support the same properties (e.g., scaling, translation, and shifting). Because DTW is the most prominent shape-based distance measure [19, 81], the majority of the k-means approaches have proposed new centroid computation methods to be used in conjunction with DTW [32, 53, 59, 64]. k-DBA has been shown to be the most robust of these approaches [64]. Another approach worth mentioning is KSC [87], which focuses on a different shape-based distance measure that offers simultaneously pairwise scaling and shifting of time series. Unfortunately, the effectiveness of such pairwise scaling and shifting, and, hence, KSC, has not been established beyond a limited number of datasets.

For completeness, we note that [28] used cross-correlation as distance measure and the arithmetic mean property for centroid computation for fuzzy clustering of fMRI data. (In Section 5 we showed that k-AVG+SBD is not competitive for our non-fuzzy setting.) Finally, cross-correlation was used to transform fMRI data into features for clustering [31], as well as for stream mining, pattern extraction, and time-series monitoring [61, 73, 85, 90].

7. CONCLUSIONS

We presented k-Shape, a partitional clustering algorithm that preserves the shapes of time series. k-Shape compares time series efficiently and computes centroids effectively under the scaling and shift invariances. Our extensive evaluation shows that k-Shape outperforms all state-of-the-art partitional, hierarchical, and spectral clustering approaches, with only one method achieving similar performance. Interestingly, this method is two orders of magnitude slower than k-Shape and its distance measure requires tuning, unlike that for k-Shape. Overall, k-Shape is a domain-independent, accurate, and scalable approach for time-series clustering.

Acknowledgments

We thank Or Biran, Eamonn Keogh, Kathy McKeown, Taesun Moon, and Kapil Thadani for invaluable discussions and feedback. We also thank Ken Ross for sharing computing resources for our experiments. This research was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D11PC20153. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government. This material is also based upon work supported by a generous gift from Microsoft Research. John Paparrizos is an Alexander S. Onassis Public Benefit Foundation Scholar.

8. REFERENCES

[1] The UCR Time Series Classification/Clustering Homepage. `http://www.cs.ucr.edu/~eamonn/time_series_data`. Accessed: May 2014.

[2] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *FODO*, pages 69–84, 1993.

[3] D. Aloise, A. Deshpande, P. Hansen, and P. Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009.

[4] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic. Discovering clusters in motion time-series data. In *CVPR*, pages 375–381, 2003.

[5] A. J. Bagnall and G. J. Janacek. Clustering time series from ARMA models with clipped data. In *KDD*, pages 49–58, 2004.

[6] Z. Bar-Joseph, G. Gerber, D. K. Gifford, T. S. Jaakkola, and I. Simon. A new approach to analyzing gene expression time series data. In *RECOMB*, pages 39–48, 2002.

[7] G. E. Batista, E. J. Keogh, O. M. Tataw, and V. M. de Souza. CID: An efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*, pages 1–36, 2013.

[8] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *AAAI Workshop on KDD*, pages 359–370, 1994.

[9] Y. Cai and R. Ng. Indexing spatio-temporal trajectories with Chebyshev polynomials. In *SIGMOD*, pages 599–610, 2004.

[10] F. Chan, A. Fu, and C. Yu. Haar wavelets for efficient similarity search of time-series: with and without time warping. *TKDE*, 15(3):686–705, 2003.

[11] L. Chen and R. Ng. On the marriage of Lp-norms and edit distance. In *VLDB*, pages 792–803, 2004.

[12] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, pages 491–502, 2005.

[13] Q. Chen, L. Chen, X. Lian, Y. Liu, and J. X. Yu. Indexable PLA for efficient similarity search. In *VLDB*, pages 435–446, 2007.

[14] Y. Chen, M. A. Nascimento, B. C. Ooi, and A. K. Tung. Spade: On shape-based pattern detection in streaming time series. In *ICDE*, pages 786–795, 2007.

[15] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[16] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In *KDD*, pages 16–22, 1998.

[17] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[18] E. Dimitriadou, A. Weingessel, and K. Hornik. A combination scheme for fuzzy clustering. *International Journal of Pattern Recognition and Artificial Intelligence*, 16(07):901–912, 2002.

[19] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: Experimental comparison of representations and distance measures. *PVLDB*, 1(2):1542–1552, 2008.

[20] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, pages 419–429, 1994.

[21] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176–190, 2008.

[22] E. Frentzos, K. Gratsias, and Y. Theodoridis. Index-based most similar trajectory search. In *ICDE*, pages 816–825, 2007.

[23] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701, 1937.

[24] M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[25] M. Gavrilov, D. Anguelov, P. Indyk, and R. Motwani. Mining the stock market: Which measure is best? In *KDD*, pages 487–496, 2000.

[26] R. Giusti and G. E. Batista. An empirical comparison of dissimilarity measures for time series classification. In *BRACIS*, pages 82–88, 2013.

[27] S. Goddard, S. K. Harms, S. E. Reichenbach, T. Tadesse, and W. J. Waltman. Geospatial decision support for drought risk management. *Communications of the ACM*, 46(1):35–37, 2003.

[28] X. Golay, S. Kollias, G. Stoll, D. Meier, A. Valavanis, and P. Boesiger. A new correlation-based fuzzy logic clustering algorithm for fMRI. *Magnetic Resonance in Medicine*, 40(2):249–260, 1998.

[29] D. Q. Goldin and P. C. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In *CP*, pages 137–153, 1995.

[30] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[31] C. Goutte, P. Toft, E. Rostrup, F. Å. Nielsen, and L. K. Hansen. On clustering fMRI time series. *NeuroImage*, 9(3):298–310, 1999.

[32] L. Gupta, D. L. Molfese, R. Tammana, and P. G. Simos. Nonlinear alignment and averaging for estimating the evoked potential. *IEEE Transactions on Biomedical Engineering*, 43(4):348–356, 1996.

[33] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.

[34] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., 3rd edition, 2011.

[35] P. Hansen and B. Jaumard. Cluster analysis and mathematical programming. *Mathematical Programming*, 79(1-3):191–215, 1997.

[36] R. Honda, S. Wang, T. Kikuchi, and O. Konishi. Mining of moving objects from time-series images and its application to satellite weather imagery. *Journal of Intelligent Information Systems*, 19(1):79–93, 2002.

[37] B. Hu, Y. Chen, and E. Keogh. Time series classification under more realistic assumptions. In *SDM*, pages 578–586, 2013.

[38] K. Kalpakis, D. Gada, and V. Puttagunta. Distance measures for effective clustering of ARIMA time-series. In *ICDM*, pages 273–280, 2001.

[39] Y. Katznelson. *An introduction to harmonic analysis*. Cambridge University Press, 2004.

[40] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: An introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[41] E. Keogh. A decade of progress in indexing and mining large time series databases. In *VLDB*, pages 1268–1268, 2006.

[42] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. In *SIGMOD*, pages 151–162, 2001.

[43] E. Keogh and J. Lin. Clustering of time-series subsequences is meaningless: Implications for previous and future research. *Knowledge and Information Systems*, 8(2):154–177, 2005.

[44] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2005.

[45] C. Kin-pong and F. Ada. Efficient time series matching by wavelets. In *ICDE*, pages 126–133, 1999.

[46] F. Korn, H. V. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *SIGMOD*, pages 289–300, 1997.

[47] C.-S. Li, P. S. Yu, and V. Castelli. MALM: A framework for mining sequence database at multiple abstraction levels. In *CIKM*, pages 267–272, 1998.

[48] X. Lian, L. Chen, J. X. Yu, G. Wang, and G. Yu. Similarity match over high speed time-series streams. In *ICDE*, pages 1086–1095, 2007.

[49] J. Lin, M. Vlachos, E. Keogh, and D. Gunopulos. Iterative incremental clustering of time series. In *EDBT*, pages 106–122, 2004.

[50] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *BSMSP*, pages 281–297, 1967.

[51] M. Mahajan, P. Nimbhorkar, and K. Varadarajan. The planar k-means problem is NP-hard. In *WALCOM*, pages 274–285. 2009.

[52] R. N. Mantegna. Hierarchical structure in financial markets. *The European Physical Journal B-Condensed Matter and Complex Systems*, 11(1):193–197, 1999.

[53] W. Meesrikamolkul, V. Niennattrakul, and C. A. Ratanamahatana. Shape-based clustering for time series data. In *PAKDD*, pages 530–541. 2012.

[54] V. Megalooikonomou, Q. Wang, G. Li, and C. Faloutsos. A multiresolution symbolic representation of time series. In *ICDE*, pages 668–679, 2005.

[55] M. D. Morse and J. M. Patel. An efficient and accurate method for evaluating time series similarity. In *SIGMOD*, pages 569–580, 2007.

[56] A. Mueen, E. Keogh, and N. Young. Logical-shapelets: An expressive primitive for time series classification. In *KDD*, pages 1154–1162, 2011.

[57] P. Nemenyi. *Distribution-free Multiple Comparisons*. PhD thesis, Princeton University, 1963.

[58] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2002.

[59] V. Niennattrakul and C. A. Ratanamahatana. Shape averaging under time warping. In *ECTI-CON*, pages 626–629, 2009.

[60] T. Oates. Identifying distinctive subsequences in multivariate time series by clustering. In *KDD*, pages 322–326, 1999.

[61] S. Papadimitriou, J. Sun, and P. S. Yu. Local correlation tracking in time series. In *ICDM*, pages 456–465, 2006.

[62] P. Papapetrou, V. Athitsos, M. Potamias, G. Kollios, and D. Gunopulos. Embedding-based subsequence matching in time-series databases. *TODS*, 36(3):17, 2011.

[63] F. Petitjean and P. Gançarski. Summarizing a set of time series by averaging: From Steiner sequence to compact multiple alignment. *Theoretical Computer Science*, 414(1):76–91, 2012.

[64] F. Petitjean, A. Ketterlin, and P. Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.

[65] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *KDD*, pages 262–270, 2012.

[66] T. Rakthanmanon, E. J. Keogh, S. Lonardi, and S. Evans. Time series epenthesis: Clustering time series streams requires ignoring some data. In *ICDM*, pages 547–556, 2011.

[67] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[68] C. A. Ratanamahatana and E. Keogh. Making time-series classification more accurate using learned constraints. In *SDM*, pages 11–22, 2004.

[69] J. Rice. *Mathematical statistics and data analysis*. Cengage Learning, 2006.

[70] E. J. Ruiz, V. Hristidis, C. Castillo, A. Gionis, and A. Jaimes. Correlating financial time series with micro-blogging activity. In *WSDM*, pages 513–522, 2012.

[71] N. Saito. *Local Feature Extraction and Its Applications Using a Library of Bases*. PhD thesis, Yale University, 1994.

[72] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978.

[73] Y. Sakurai, S. Papadimitriou, and C. Faloutsos. Braid: Stream mining through group lag correlations. In *SIGMOD*, pages 599–610, 2005.

[74] Y. Shou, N. Mamoulis, and D. Cheung. Fast and exact warping of time series using adaptive segmental approximations. *Machine Learning*, 58(2-3):231–267, 2005.

[75] A. Stefan, V. Athitsos, and G. Das. The move-split-merge metric for time series. *TKDE*, 25(6):1425–1438, 2013.

[76] K. Uehara and M. Shimada. Extraction of primitive motion and discovery of association rules from human motion data. In *Progress in Discovery Science*, pages 338–348. 2002.

[77] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh. Indexing multidimensional time-series. *The VLDB Journal*, 15(1):1–20, 2006.

[78] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *ICDE*, pages 673–684, 2002.

[79] H. Wang, Y. Cai, Y. Yang, S. Zhang, and N. Mamoulis. Durable queries over historical time series. *TKDE*, 26(3):595–607, 2014.

[80] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348, 1994.

[81] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013.

[82] X. Wang, K. Smith, and R. Hyndman. Characteristic-based clustering for time series data. *Data Mining and Knowledge Discovery*, 13(3):335–364, 2006.

[83] T. Warren Liao. Clustering of time series data - a survey. *Pattern Recognition*, 38(11):1857–1874, 2005.

[84] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, pages 80–83, 1945.

[85] D. Wu, Y. Ke, J. X. Yu, S. Y. Philip, and L. Chen. Detecting leaders from correlated time series. In *DASFAA*, pages 352–367, 2010.

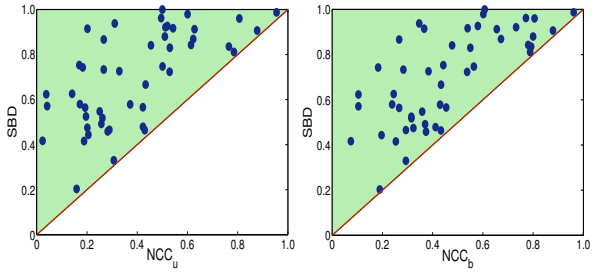
[86] Y. Xiong and D.-Y. Yeung. Mixtures of ARMA models for model-based time series clustering. In *ICDM*, pages 717–720, 2002.

[87] J. Yang and J. Leskovec. Patterns of temporal variation in online media. In *WSDM*, pages 177–186, 2011.

[88] L. Ye and E. Keogh. Time series shapelets: A new primitive for data mining. In *KDD*, pages 947–956, 2009.

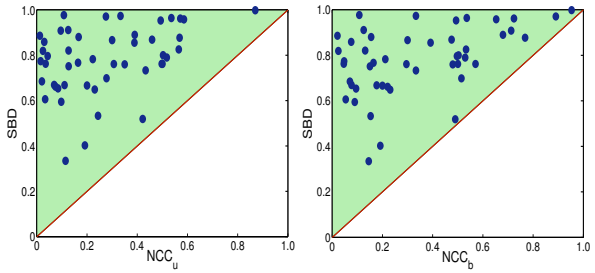
[89] J. Zakaria, A. Mueen, and E. Keogh. Clustering time series using unsupervised-shapelets. In *ICDM*, pages 785–794, 2012.

[90] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, pages 358–369, 2002.



(a) SBD vs. NCC_u (b) SBD vs. NCC_b

Figure 10: Comparison of SBD, NCC_b, and NCC_u over 48 datasets under the *OptimalScaling* normalization. Circles above the diagonal indicate datasets over which SBD has better accuracy than the compared method.



(a) SBD vs. NCC_u (b) SBD vs. NCC_b

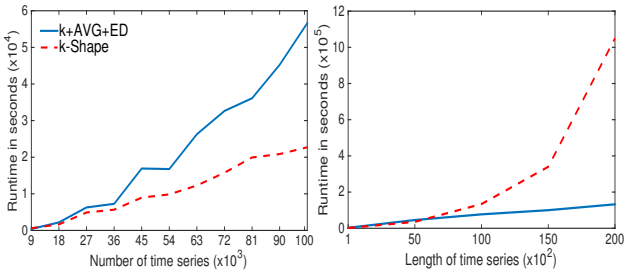
Figure 11: Comparison of SBD, NCC_b, and NCC_u over 48 datasets under the *ValuesBetween0-1* normalization. Circles above the diagonal indicate datasets over which SBD has better accuracy than the compared method.

APPENDIX

A. CROSS-CORRELATION VARIANTS UNDER TIME-SERIES NORMALIZATIONS

As discussed in Section 3.1, the choice of normalizations for the data and the cross-correlation measure can significantly impact the cross-correlation sequence produced. To understand the sensitivity of cross-correlation variants to data normalizations we evaluate their performance using all 48 datasets from the UCR time-series collection [1]. This collection contains z-normalized datasets but the unnormalized versions are not available. Therefore, to experiment with unnormalized versions of these datasets and ensure that the initial sequences differ in amplitude, we first multiply each sequence with a random number chosen individually for that sequence. Then, we perform and study three common time-series normalizations: (1) *OptimalScaling*: to match two time series $\vec{x}$ and $\vec{y}$, we compute their optimal scaling coefficient $c = \frac{\vec{x} \cdot \vec{y}^T}{\vec{y} \cdot \vec{y}^T}$, which is used for every pairwise comparison (e.g., SBD($\vec{x}$,$\vec{y}$) is computed as SBD($\vec{x}$,$c \cdot \vec{y}$)); (2) *ValuesBetween0-1*: we normalize each sequence $\vec{x}$ such that its values fall between 0 and 1, by transforming it into $\vec{x}' = \frac{\vec{x} - min(\vec{x})}{max(\vec{x}) - min(\vec{x})}$; and (3) *z-normalization*: we normalize each sequence $\vec{x}$ such that its mean is 0 and its standard deviation is 1, by transforming it into $\vec{x}' = \frac{\vec{x} - mean(\vec{x})}{std(\vec{x})}$.

We evaluate the 1-NN classification accuracy of each cross-correlation variant, namely, SBD, NCC_u, and NCC_b, on each dataset for the three common time-series normalization scenarios. Figure 10 presents the results for the *OptimalScaling* normalization. In particular, SBD significantly outperforms both NCC_u (Figure 10a) and NCC_b (Figure 10b) in



(a) Varying n for m=128 (b) Varying m for n=18K

Figure 12: Runtime of k-AVG+ED and k-Shape as a function of (a) the number of time series n and (b) the time series length m.

all 48 datasets, and, in turn, NCC_b outperforms NCC_u in 40 datasets, with statistical significance.

Similarly, for *ValuesBetween0-1* data normalization, SBD significantly outperforms NCC_u (Figure 11a) and NCC_b (Figure 11b) in all 48 datasets. NCC_b performs similarly to or better than NCC_u in 41 datasets and the Wilcoxon test suggests this difference in accuracy is statistically significant. Finally, for *z-normalization*, SBD and NCC_b achieve similar performance and both significantly outperform NCC_u.

Therefore, we can conclude that SBD is the most robust cross-correlation variant as it achieves better performance across multiple different time series normalizations. On average, SBD achieves accuracy values of 0.699, 0.779, and 0.795, for *OptimalScaling*, *ValuesBetween0-1*, and *z-normalization* normalizations, respectively.

B. SCALABILITY OF K-SHAPE

As discussed in Section 3.3, k-Shape requires $\mathcal{O}(\max\{n \cdot k \cdot m \cdot \log(m), n \cdot m^2, k \cdot m^3\})$ time per iteration to cluster n time series of length m into k clusters. The majority of the computational cost of our algorithm depends on the time-series length m, whereas its dependence on the number n of time series is linear. To better understand the scalability of k-Shape, we use the synthetic CBF dataset [71] because it enables experiments with varying values of both n and m without changing any of its general properties. Figure 12 reports the results of our scalability experiments where both n and m are up to one order of magnitude larger than the biggest dataset in the UCR archive [1]. We report the average CPU runtime of 5 runs. Specifically, in Figure 12a, we vary n while we set $m = 128$ as in the original CBF dataset: k-Shape, similarly to k+AVG+ED, scales linearly with the number of time series without any loss in accuracy for both methods. Importantly, k-Shape, remains significantly faster than k-AVG+ED with the increasing number of time series because k-Shape requires 45% fewer iterations to converge than k+AVG+ED does. Then, in Figure 12b, we vary m while we set $n = 18K$: k-Shape still outperforms k-AVG+ED for $m \ll n$ but k-Shape becomes slower, as expected, when the time-series length is large and approaches the total number of time series in the dataset. Similarly to the previous experiment, there is no loss in accuracy for either technique.